REMARKS

Claims 1, 7, 11-13, 14, and 24-28 have been amended; claims 10 and 23 have been cancelled; and new claims 29-30 have been added. Claims 1-9, 11-22 and 24-30 are pending.

Figure 4 has been amended as requested in the Office Action.

The specification has been amended to address informalities noted in the Office Action. It is noted that the original reference to "communications channel 171" on page 20, line 14 was a typographical error in the text, and has been corrected to read "communications channel 170-1" which is shown in Figure 1. Thus, Figure 1 requires no correction, and none has been made.

An excerpt from an article entitled "The Transmission Control Protocol" found at http://condor.depaul.edu/~jkristof/technotes/tcp.html is included herewith. The Examiner is referred to pages 2 and 4 of this article for an explanation of the terms "segment", "SYN" and "FIN". It is hoped that this material provides the clarification requested in the Office Action.

Claim 26 has been amended to correctly depend from apparatus claim 14 rather than method claim 1.

The original independent claims 1, 14 and 27-28 have all been amended to incorporate the features of dependent claims 10 and 23, which have been cancelled. It is believed that these claims clearly distinguish over the art of record, as discussed below.

Additionally, new independent claims 29-30 have been added. These claims recite additional features not recited in the original independent claims. In particular, the method/apparatus: (i) communicates with a separate network communications device, such as a router; (ii) executes the various steps with respect to allocated bandwidth, which as generally known is a channel-imposed limit on transmission rate and not a transmission rate itself; (iii) operates specifically in response to "activity of a network application indicating that a new communications session will be established", examples of which include session initiation commands, http gets, embedded URLs etc. as described in the

application as filed; and (iv) negotiates with the NCD to obtain the desired bandwidth. The significance of these features will be clear in light of the discussion of the Ravi reference below.

In the Office Action, claims 1-12, 14-25 and 27-28 are rejected under 35 U.S.C. § 102(e) as being anticipated by Ravi et al., US 6,292,834 B1 ("Ravi"). This rejection is respectfully traversed with respect to the claims as amended.

Claim 1 as amended herein recites a method in a browser-enabled communications device for dynamically adjusting bandwidth of a communications channel. The method includes:

...

> detecting a first event indicating a first anticipated change in a bandwidth requirement of the communications channel, the first event indicating a browser event requiring a browser in the communications device to access content from a remote computer system;
>
> calculating a first new value for a bandwidth setting of the communications channel in response to detecting the first event, the first new value of the bandwidth setting of the communications channel being calculated to accommodate additional bandwidth used to receive the content from the remote computer system; and
>
> adjusting a bandwidth characteristic of the communications channel according to the first new value of the bandwidth setting such that communications channel can accommodate the first anticipated change in the bandwidth requirement.

As described in the application, this method enables a browser-enabled device such as a client in a client-server system to improve performance by adjusting the amount of network bandwidth allocated to the channel between the client and the server, based on what are referred to as "browser events". Examples of such browser events in the specification and claims include parsing content accessed by the browser to detect a reference (claim 11; examples

include a URL), and detecting a communications session message generated by the browser on the network (claim 12; examples include a TCP "SYN" message). It is noted that the method is concerned with the bandwidth setting of the channel, not the transmission rate of the server. Thus, the claimed method concerns the problem of browser performance being limited not by the transmission rate of the server, but rather by the maximum data rate of the channel as may be enforced by a network communications device, for example. If the client and the server are both capable of operating at a data transfer rate of X, for example, this operating capability is wasted insofar as it exceeds an allocated channel bandwidth of Y. By the same token, the data transfer capacity of the network is wasted insofar as Y exceed X. The method of claim 1 operates to adjust Y on the assumption that the server will automatically adjust its actual transmission rate, and the client its actual reception rate, accordingly.

Ravi teaches a technique in which a client computer includes a playout buffer for playing out a stream of media data received from a server, and the transmission rate from the server is dynamically matched to the available bandwidth capacity of the network connection between the server and the client computer. If a playtime of the playout buffer, which is one measure of the number of data packets currently in the playout buffer, drops below a dynamically computed Decrease_Bandwidth (DEC_BW) threshold, then the transmission rate is decreased by sending a DEC_BW message to the server. Conversely, if the number of packets remaining in the playout buffer rises above a dynamically computed Upper Increase_Bandwidth (INC_BW) threshold and does not drop below a Lower INC_BW threshold for at least an INC_BW wait period, then the transmission rate is incremented. The transmission rate can be selected from among a predetermined set of discrete bandwidth values or from within a continuous range of bandwidth values.

Although Ravi makes unfortunate use of the term "bandwidth", it is explicitly explained that what is meant by that term is actually "transmission rate" (see col. 7 lines 1-2). Moreover, as stated above, Ravi's technique adjusts

transmission rate from the server so as to match it with the bandwidth capacity of the network connection, which is implicitly assumed to be beyond the control of the technique. That is, in Ravi's technique, there is no attempt to control the amount of bandwidth allocated to the channel; it is assumed to be fixed or at least not controllable. Rather, Ravi's technique operates to control the transmission rate from the server to match it with the available channel bandwidth, so that neither the allocated bandwidth nor the processing capabilities of the client or server are unnecessarily wasted. Based on this description, Ravi's technique can be seen as a classic flow-control technique that operates to make efficient use of a predetermined (or at least uncontrolled) channel bandwidth.

Additionally, Ravi is seen to make adjustments in transmission rate based on the values of "performance variables" pertaining to the playout buffer (i.e., various threshold values, playtimes, changes in playtime, and other related variables), and not in response to events that require a browser to access content from a remote computer system. It is noted in this respect that col. 6, lines 31-33 of Ravi, which are referred to in the Office Action, describe only that Ravi's technique is directed to the streaming of packets from a stream server to a client computer. This is done by use of the "performance variables", which are hardware-level variables that provide information about the flow of data into and out of the playout buffer when data is being streamed from the server. Adjustments to transmission rate during a particular streaming operation are made in response to playout buffer conditions encountered during the same streaming operation. The technique has no knowledge of how the streaming operation was initiated, for example, nor would any such knowledge be helpful in Ravi. In particular, the values of the performance variables do not reflect any conditions that require Ravi's browser to access content from the server – they simply provide information about the playout buffer during a particular streaming operation.

It is respectfully submitted that Ravi does not anticipate claim 1 under 35 U.S.C. § 102(e), because it fails to teach all the elements thereof. In fact, Ravi does not teach any of the above-highlighted elements of claim 1. Ravi's technique operates to adjust the transmission rate from a server, on the assumption that the uncontrolled channel has a corresponding capacity. This is in contrast to the technique of claim 1, that operates to (i) detect an event indicating a first anticipated change in a bandwidth requirement of the communications channel, (ii) calculate a new value for a bandwidth setting of the communications channel in response to detecting the first event, and (iii) adjust a bandwidth characteristic of the communications channel according to the first new value of the bandwidth setting such that communications channel can accommodate the first anticipated change in the bandwidth requirement.

Moreover, the transmission-rate adjustments in Ravi are made in response to the values of performance variables reflecting operational aspects of a playout buffer, and not in response to a browser event requiring a browser in the communications device to access content from a remote computer system, as set forth in claim 1. Accordingly, Ravi is not seen to teach the above elements of claim 1, and therefore cannot anticipate claim 1 under 35 U.S.C. § 102(e).

It will be appreciated that claims 2-9, 11-22, and 24-28 incorporate, either directly or indirectly, the same features of claim 1 discussed above, and accordingly are allowable for at least the same reasons.

New claims 29-30 include the following features that are not believed to be shown in Ravi or the other art of record:

...

detecting activity of the network application indicating that a new communications session will be established requiring a first anticipated change in bandwidth allocated to the communications channel;

calculating a first new value for an allocated bandwidth setting of the communications channel in response to detecting the network application activity; and

negotiating with the network communications device to adjust a bandwidth characteristic of the communications channel according to the first new value of the allocated bandwidth setting to effect the first anticipated change in the bandwidth allocated to the communications channel.

The above elements employ the term "bandwidth allocated to the communications channel" to describe what is being adjusted. This further clarifies that the method is adjusting a parameter of the channel and not of the transmitter.

Furthermore, these claims include an element of negotiating with a network communications device to adjust the bandwidth characteristic of the channel. It should be clear that no such element is present in Ravi. In Ravi, the messages generated by the client are sent to the transmitting server, not to any network communications devices that provide the channel over which the client and the server are communicating. Thus, it is believed that new claims 29 and 30 are even further distinguished from Ravi, and accordingly are allowable in view thereof.

A petition and fee for a one-month extension of time is submitted herewith. If the U.S. Patent and Trademark Office deems any additional fees necessary, such fees may be charged to the account of the undersigned, Deposit Account No. 50-0901.

If the enclosed papers or fees are considered incomplete, the Patent Office is respectfully requested to contact the undersigned attorney.

Respectfully submitted,

James F. Thompson
Attorney for Applicant(s)
Registration No.: 36,699
CHAPIN & HUANG, L.L.C.
Westborough Office Park
1700 West Park Drive
Westborough, Massachusetts 01581
Telephone: (508) 366-9600
Facsimile: (508) 616-9805
Customer No.: 022468

Attorney Docket No.: CIS00-3487

Dated: January 19, 2005

# The Transmission Control Protocol


## Abstract

It is important to understand TCP if one is to understand the historic, current and future architecture of the Internet protocols. Most applications on the Internet make use of TCP, relying upon its mechanisms that ensure safe delivery of data across an unreliable IP layer below. In this paper we explore the fundamental concepts behind TCP and how it is used to transport data between two endpoints.


## 1. Introduction

The Transmission Control Protocol (TCP) standard is defined in the Request For Comment (RFC) standards document number 793 [10] by the Internet Engineering Task Force (IETF). The original specification written in 1981 was based on earlier research and experimentation in the original ARPANET. The design of TCP was heavily influenced by what has come to be known as the "end-to-end argument" [3].

As it applies to the Internet, the end-to-end argument says that by putting excessive intelligence in physical and link layers to handle error control, encryption or flow control you unnecessarily complicate the system. This is because these functions will usually need to be done at the endpoints anyway, so why duplicate the effort along the way? The result of an end-to-end network then, is to provide minimal functionality on a hop-by-hop basis and maximal control between end-to-end communicating systems.

The end-to-end argument helped determine how two characteristics of TCP operate; performance and error handling. TCP performance is often dependent on a subset of algorithms and techniques such as flow control and congestion control. Flow control determines the rate at which data is transmitted between a sender and receiver. Congestion control defines the methods for implicitly interpreting signals from the network in order for a sender to adjust its rate of transmission.

The term congestion control is a bit of a misnomer. Congestion avoidance would be a better term since TCP cannot control congestion per se. Ultimately intermediate devices, such as IP routers would only be able to control congestion.

Congestion control is currently a large area of research and concern in the network community. A companion study on congestion control examines the current state of activity in that area [9].

Timeouts and retransmissions handle error control in TCP. Although delay could be substantial, particularly if you were to implement real-time applications, the use of both techniques offer error detection and error correction thereby guaranteeing that data will eventually be sent successfully.

The nature of TCP and the underlying packet switched network provide formidable challenges for managers, designers and researchers of networks. Once regulated to low speed data communication applications, the Internet and in part TCP are being used to support very high speed communications of voice, video and data. It is unlikely that the Internet protocols will remain static as the applications change and expand. Understanding the current state of affairs will assist us in understanding protocol changes made to support future applications.

### 1.1 Transmission Control Protocol

TCP is often described as a byte stream, connection-oriented, reliable delivery transport layer protocol. In

turn, we will discuss the meaning for each of these descriptive terms.

### 1.1.1 Byte Stream Delivery

TCP interfaces between the application layer above and the network layer below. When an application sends data to TCP, it does so in 8-bit byte streams. It is then up to the sending TCP to segment or delineate the byte stream in order to transmit data in manageable pieces to the receiver[1]. It is this lack of 'record boundaries" which give it the name "byte stream delivery service".

### 1.1.2 Connection-Oriented

Before two communicating TCPs can exchange data, they must first agree upon the willingness to communicate. Analogous to a telephone call, a connection must first be made before two parties exchange information.

### 1.1.3 Reliability

A number of mechanisms help provide the reliability TCP guarantees. Each of these is described briefly below.

*Checksums*. All TCP segments carry a checksum, which is used by the receiver to detect errors with either the TCP header or data.

*Duplicate data detection*. It is possible for packets to be duplicated in packet switched network; therefore TCP keeps track of bytes received in order to discard duplicate copies of data that has already been received.[2]

*Retransmissions*. In order to guarantee delivery of data, TCP must implement retransmission schemes for data that may be lost or damaged. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period (see timers below) calls for a retransmission.

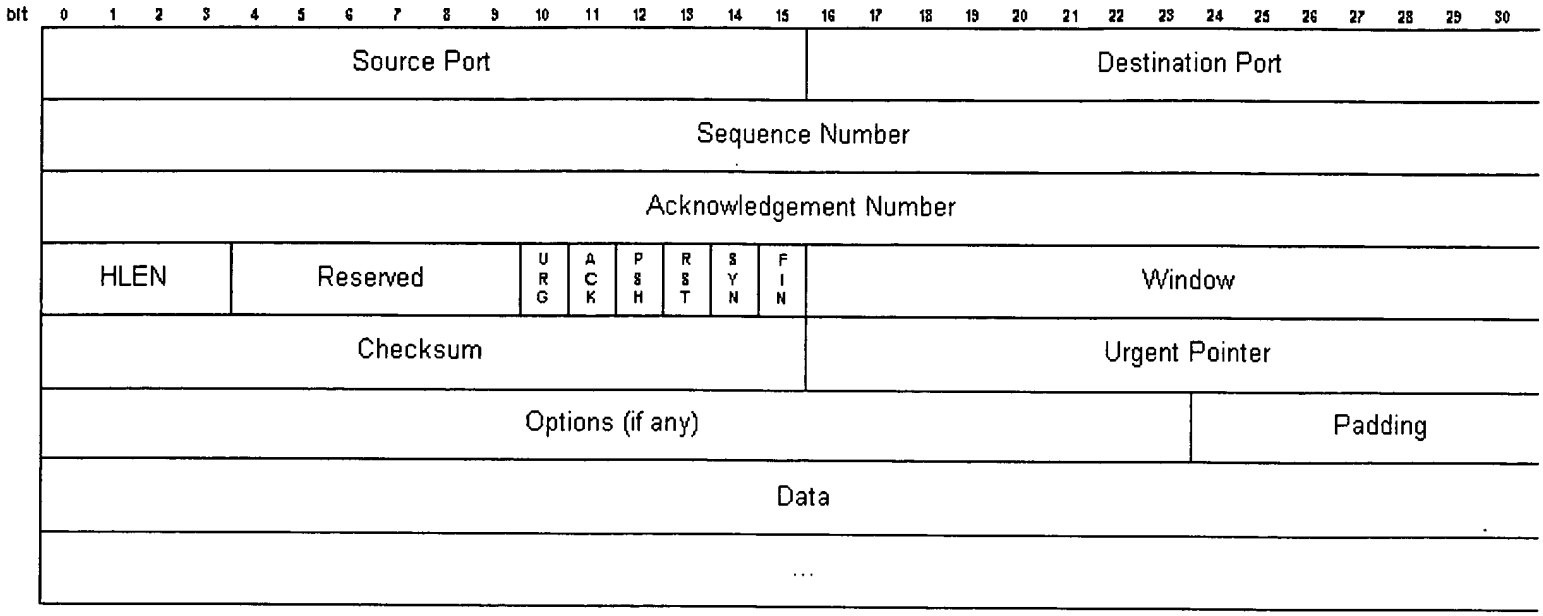
*Sequencing*. In packet switched networks, it is possible for packets to be delivered out of order. It is TCP's job to properly sequence segments it receives so it can deliver the byte stream data to an application in order.

*Timers*. TCP maintains various static and dynamic timers on data sent. The sending TCP waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving an acknowledgement, the sender can retransmit the segment.

## 1.2 TCP Header Format

Remember that the combination of TCP header and TCP in one packet is called a TCP segment. Figure 1 depicts the format of all valid TCP segments. The size of the header without options is 20 bytes. We will briefly define each field of the TCP header below.

**Figure 1 - TCP Header Format**




### 1.2.1 Source Port

A 16-bit number identifying the application the TCP segment originated from within the sending host. The port numbers are divided into three ranges, well-known ports (0 through 1023), registered ports (1024 through 49151) and private ports (49152 through 65535). Port assignments are used by TCP as an interface to the application layer. For example, the TELNET server is always assigned to the well-known port 23 by default on TCP hosts. A complete pair of IP addresses (source and destination) plus a complete pair of TCP ports (source and destination) define a single TCP connection that is globally unique. See [5] for further details.

### 1.2.2 Destination Port

A 16-bit number identifying the application the TCP segment is destined for on a receiving host. Destination ports use the same port number assignments as those set aside for source ports [5].

### 1.2.3 Sequence Number

A 32-bit number identifying the current position of the first data byte in the segment within the entire byte stream for the TCP connection. After reaching $2^{32}$ -1, this number will wrap around to 0.

### 1.2.4 Acknowledgement Number

A 32-bit number identifying the next data byte the sender expects from the receiver. Therefore, the number will be one greater than the most recently received data byte. This field is only used when the ACK control bit is turned on (see below).

### 1.2.5 Header Length

A 4-bit field that specifies the total TCP header length in 32-bit words (or in multiples of 4 bytes if you prefer). Without options, a TCP header is always 20 bytes in length. The largest a TCP header may be is 60 bytes. This field is required because the size of the options field(s) cannot be determined in advance. Note that this field is called "data offset" in the official TCP standard, but header length is more commonly used.

### 1.2.6 Reserved

A 6-bit field currently unused and reserved for future use.

### 1.2.7 Control Bits

*Urgent Pointer (URG)*. If this bit field is set, the receiving TCP should interpret the urgent pointer field (see below).

*Acknowledgement (ACK)*. If this bit field is set, the acknowledgement field described earlier is valid.

*Push Function (PSH)*. If this bit field is set, the receiver should deliver this segment to the receiving application as soon as possible. An example of its use may be to send a Control-BREAK request to an application, which can jump ahead of queued data.

*Reset the Connection (RST)*. If this bit is present, it signals the receiver that the sender is aborting the connection and all queued data and allocated buffers for the connection can be freely relinquished.

*Synchronize (SYN)*. When present, this bit field signifies that sender is attempting to "synchronize" sequence numbers. This bit is used during the initial stages of connection establishment between a sender and receiver.

*No More Data from Sender (FIN)*. If set, this bit field tells the receiver that the sender has reached the end of its byte stream for the current TCP connection.

### 1.2.8 Window

A 16-bit integer used by TCP for flow control in the form of a data transmission window size. This number tells the sender how much data the receiver is willing to accept. The maximum value for this field would limit the window size to 65,535 bytes, however a "window scale" option can be used to make use of even larger windows.

### 1.2.9 Checksum

A TCP sender computes a value based on the contents of the TCP header and data fields. This 16-bit value will be compared with the value the receiver generates using the same computation. If the values match, the receiver can be very confident that the segment arrived intact.

### 1.2.10 Urgent Pointer

In certain circumstances, it may be necessary for a TCP sender to notify the receiver of urgent data that should be processed by the receiving application as soon as possible. This 16-bit field tells the receiver when the last byte of urgent data in the segment ends.

### 1.2.11 Options

In order to provide additional functionality, several optional parameters may be used between a TCP sender and receiver. Depending on the option(s) used, the length of this field will vary in size, but it cannot be larger than 40 bytes due to the size of the header length field (4 bits). The most common option is the maximum segment size (MSS) option. A TCP receiver tells the TCP sender the maximum segment size it is willing to accept through the use of this option. Other options are often used for various flow control and congestion

control techniques.

### 1.2.12 Padding

Because options may vary in size, it may be necessary to "pad" the TCP header with zeroes so that the segment ends on a 32-bit word boundary as defined by the standard [10].

### 1.2.13 Data

Although not used in some circumstances (e.g. acknowledgement segments with no data in the reverse direction), this variable length field carries the application data from TCP sender to receiver. This field coupled with the TCP header fields constitutes a TCP segment.

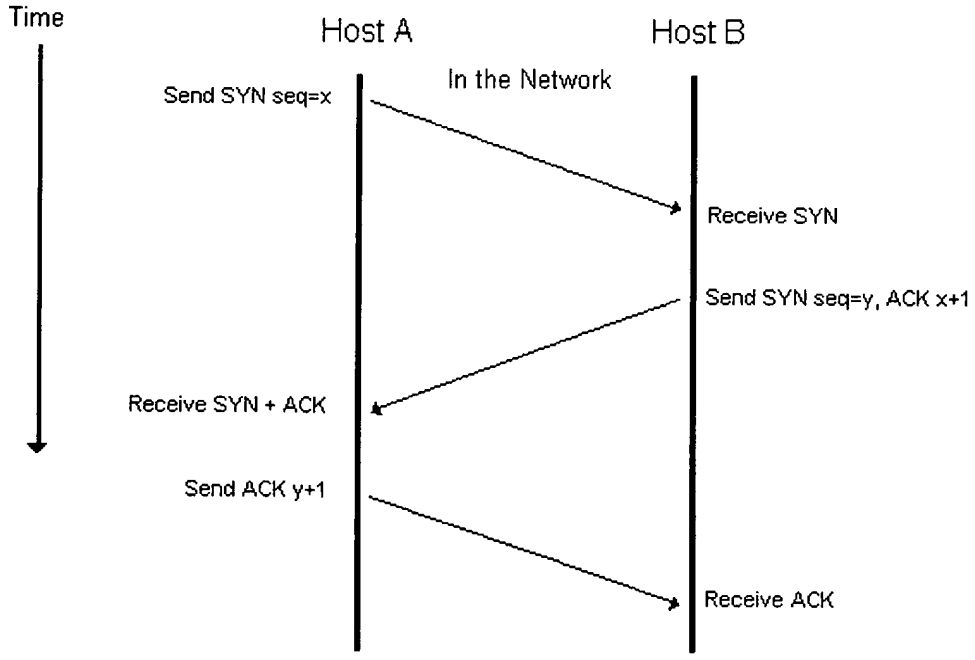
## 2. Connection Establishment and Termination

TCP provides a connection-oriented service over packet switched networks. Connection-oriented implies that there is a virtual connection between two endpoints.[3] There are three phases in any virtual connection. These are the connection establishment, data transfer and connection termination phases.

### 2.1 Three-Way Handshake

In order for two hosts to communicate using TCP they must first establish a connection by exchanging messages in what is known as the three-way handshake. The diagram below depicts the process of the three-way handshake.

**Figure 2 - TCP Connection Establishment**




From figure 2, it can be seen that there are three TCP segments exchanged between two hosts, Host A and Host B. Reading down the diagram depicts events in time.

To start, Host A initiates the connection by sending a TCP segment with the SYN control bit set and an initial sequence number (ISN) we represent as the variable x in the sequence number field.

At some moment later in time, Host B receives this SYN segment, processes it and responds with a TCP segment of its own. The response from Host B contains the SYN control bit set and its own ISN represented as variable y. Host B also sets the ACK control bit to indicate the next expected byte from Host A should contain data starting with sequence number x+1.

When Host A receives Host B's ISN and ACK, it finishes the connection establishment phase by sending a final acknowledgement segment to Host B. In this case, Host A sets the ACK control bit and indicates the next expected byte from Host B by placing acknowledgement number y+1 in the acknowledgement field.

In addition to the information shown in the diagram above, an exchange of source and destination ports to use for this connection are also included in each senders' segments.[4]

### 2.2 Data Transfer

Once ISNs have been exchanged, communicating applications can transmit data between each other. Most of the discussion surrounding data transfer requires us to look at flow control and congestion control techniques which we discuss later in this document and refer to other texts [9]. A few key ideas will be briefly made here, while leaving the technical details aside.

A simple TCP implementation will place segments into the network for a receiver as long as there is data to send and as long as the sender does not exceed the window advertised by the receiver. As the receiver accepts and processes TCP segments, it sends back positive acknowledgements, indicating where in the byte stream it is. These acknowledgements also contain the "window" which determines how many bytes the receiver is currently willing to accept. If data is duplicated or lost, a "hole" may exist in the byte stream. A receiver will continue to acknowledge the most current contiguous place in the byte stream it has accepted.

If there is no data to send, the sending TCP will simply sit idly by waiting for the application to put data into the byte stream or to receive data from the other end of the connection.

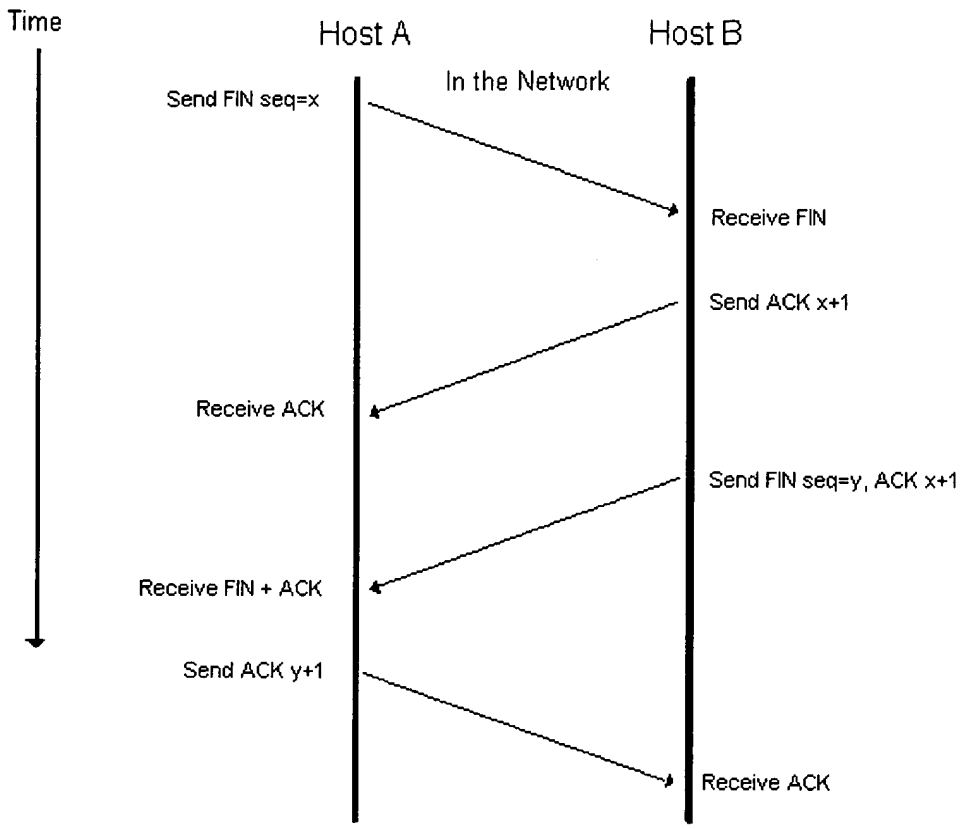
If data queued by the sender reaches a point where data sent will exceed the receiver's advertised window size, the sender must halt transmission and wait for further acknowledgements and an advertised window size that is greater than zero before resuming.

Timers are used to avoid deadlock and unresponsive connections. Delayed transmissions are used to make more efficient use of network bandwidth by sending larger "chunks" of data at once rather than in smaller individual pieces.[5]

### 2.3 Connection Termination

In order for a connection to be released, four segments are required to completely close a connection. Four segments are necessary due to the fact that TCP is a full-duplex protocol, meaning that each end must shut down independently.[6] The connection termination phase is shown in figure 3 below.

**Figure 3 - TCP Connection Termination**




Notice that instead of SYN control bit fields, the connection termination phase uses the FIN control bit fields to signal the close of a connection.

To terminate the connection in our example, the application running on Host A signals TCP to close the connection. This generates the first FIN segment from Host A to Host B. When Host B receives the initial FIN segment, it immediately acknowledges the segment and notifies its destination application of the termination request. Once the application on Host B also decides to shut down the connection, it then sends its own FIN segment, which Host A will process and respond with an acknowledgement.

## 3. Sliding Window and Flow Control

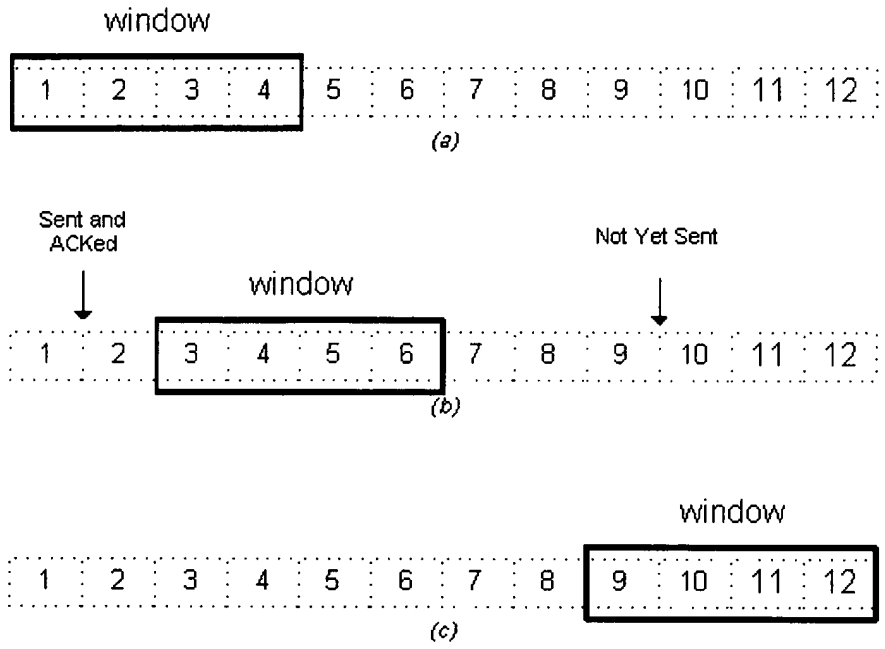
Flow control is a technique whose primary purpose is to properly match the transmission rate of sender to that of the receiver and the network. It is important for the transmission to be at a high enough rate to ensure good performance, but also to protect against overwhelming the network or receiving host.

In [8], we note that flow control is not the same as congestion control. Congestion control is primarily concerned with a sustained overload of network intermediate devices such as IP routers.

TCP uses the window field, briefly described previously, as the primary means for flow control. During the data transfer phase, the window field is used to adjust the rate of flow of the byte stream between communicating TCPs.

Figure 4 below illustrates the concept of the sliding window.

**Figure 4 - Sliding Window**




In this simple example, there is a 4-byte sliding window. Moving from left to right, the window "slides" as bytes in the stream are sent and acknowledged.[7] The size of the window and how fast to increase or decrease the window size is an area of great research. We again refer to other documents for further detail [9].

## 4. Congestion Control

TCP congestion control and Internet traffic management issues in general is an active area of research and experimentation. This final section is a very brief summary of the standard congestion control algorithms widely used in TCP implementations today. These algorithms are defined in [6] and [7]. Their use with TCP was standardized in [1].

### 4.1 Slow Start

Slow Start, a requirement for TCP software implementations is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. This is accomplished through the return rate of acknowledgements from the receiver. In other words, the rate of acknowledgements returned by the receiver determine the rate at which the sender can transmit data.

When a TCP connection first begins, the Slow Start algorithm initializes a congestion window to one segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase. When acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the transmission window.

Slow Start is actually not very slow when the network is not congested and network response time is good. For example, the first successful transmission and acknowledgement of a TCP segment increases the window to two segments. After successful transmission of these two segments and acknowledgements completes, the window is increased to four segments. Then eight segments, then sixteen segments and so on, doubling from there on out up to the maximum window size advertised by the receiver or until congestion

finally does occur.

### 4.2 Congestion Avoidance

During the initial data transfer phase of a TCP connection the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. If this happens, Congestion Avoidance is used to slow the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow.

In the Congestion Avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked (see below).

As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

### 4.3 Fast Retransmit

When a duplicate ACK is received, the sender does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgement. Typically no more than one or two duplicate ACKs should be received when simple out of order conditions exist. If however more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost. The TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fact that the receiver had enough time to send three duplicate ACKs.

When three or more duplicate ACKs are received, the sender does not even wait for a retransmission timer to expire before retransmitting the segment (as indicated by the position of the duplicate ACK in the byte stream). This process is called the Fast Retransmit algorithm and was first defined in [7]. Immediately following Fast Retransmit is the Fast Recovery algorithm.

### 4.4 Fast Recovery

Since the Fast Retransmit algorithm is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. Why? The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication that serious network congestion may not exist and that the lost segment was a rare event. So instead of reducing the flow of data abruptly by going all the way into Slow Start, the sender only enters Congestion Avoidance mode.

Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode. This allows for higher throughput under the

condition of only moderate congestion [23].

## 5. Conclusions

TCP is a fairly complex protocol that handles the brunt of functionality in a packet switched network such as the Internet. Supporting the reliable delivery of data on a packet switched network is not a trivial task. This document only scratches the surface of the TCP internals, but hopefully provided the reader with an appreciation and starting point for further interest in TCP. Even after almost 20 years of standardization, the amount of work that goes into supporting and designing reliable packet switched networks has not slowed. It is an area of great activity and there are many problems to be solved. As the Internet continues to grow, our reliance on TCP will become increasingly important. It is therefore imperative for network engineers, designers and researchers to be as well versed in the technology as possible.

---

1. The word "segment" is the term used to describe TCP's data unit size transmitted to a receiver. TCP determines the appropriate use of this segment size rather than leaving it up to higher layer protocols and applications.
2. Duplicate packets are typically caused by retransmissions, where the first packet may have been delayed and the second sent due to the lack of an acknowledgement. The receiver may then receive two identical packets.
3. As opposed to a connectionless-oriented protocol such as that used by the user datagram protocol (UDP).
4. There are additional details of the connection establishment, data transfer and termination phases that are beyond the scope of this document. For curious readers, I recommend consulting a more complete reference such as [4], [11] and of course the official standard RFC 793 [10].
5. It was discovered early on that some implementations of TCP performed poorly due to this scenario. It has been termed the silly window syndrome and documented in [2].
6. Although it is possible, it is not very common for TCP to be operating in the "half-close state". See [11] for further details.
7. We assume in this example that bytes are immediately acknowledged so that the window can move forward. In practice the sender's window shrinks and grows dynamically as acknowledgements arrive in time.

### Abbreviations

| | |
|---|---|
| ***ACK*** | Acknowledgement |
| ***bit*** | binary digit |
| ***IETF*** | Internet Engineering Task Force |
| ***IP*** | Internet Protocol |
| ***ISN*** | Initial Sequence Number |
| ***RFC*** | Request For Comments |
| ***TCP*** | Transmission Control Protocol |
| ***TCP/IP*** | Transmission Control Protocol/Internet Protocol |
| ***UDP*** | User Datagram Protocol |

## References


[1] Robert Braden. Requirements for Internet Hosts - Communication Layers, October 1989, RFC 1122.

[2] David D. Clark. Window Acknowledgement and Strategy in TCP, July 1982, RFC 813.

[3] David D. Clark. The Design Philosophy of the DARPA Internet Protocols. In Proceedings SIGCOMM '88, Computer Communications Review Vol. 18, No. 4, August 1988, pp. 106-114).

[4] Douglas E. Comer. Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture. Prentice Hall, ISBN: 0-13-216987-8. March 24, 1995.

[5] Internet Assigned Numbers Authority. Port Number Assignment, February 2000.

[6] Van Jacobson. Congestion Avoidance and Control. Computer Communications Review, Volume 18 number 4, pp. 314-329, August 1988.

[7] Van Jacobson. Modified TCP Congestion Control Avoidance Algorithm. end-2-end-interest mailing list, April 30, 1990.

[8] S. Keshav. An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network. Addison Wesley, ISBN: 0-201-63442-2. July, 1997.

[9] John Kristoff. TCP Congestion Control, March 2000.

[10] Jon Postel. Transmission Control Protocol, September 1981, RFC 793.

[11] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley, ISBN: 0-201-63346-9. January 1994.


---

Goto John Kristoff's Home Page

*Last updated: April 24, 2000*

IN THE DRAWING

Please replace the originally filed drawings with the replacement drawings included herewith. These replacement drawings include the following changes:

In Figure 4, hash lines and dotted lines have been added to more clearly show the correspondence between each of the times T0 – T5 and the events that occur at such times.